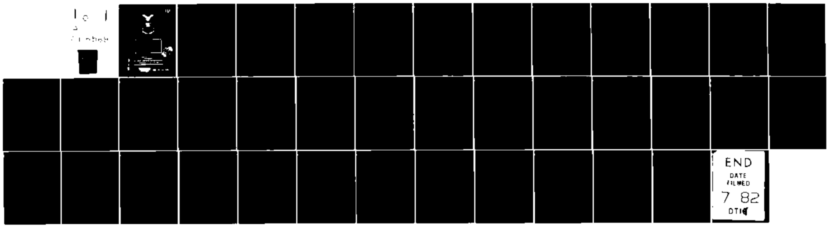
AD-A115 569 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 12/1
A NEW METHOD OF SOLVING ILL-CONDITIONED SYSTEMS OF EQUATIONS.(U)
DEC 81 B S BIRMINGHAM
UNCLASSIFIED AFIT/GA/MA/81D-1 NL



AD A115569

[illegible]

AFIT/GA/MA/81D-1

A NEW METHOD OF SOLVING ILL-CONDITIONED SYSTEMS OF EQUATIONS

THESIS

AFIT/GA/MA/81D-1 Brian S. Birmingham
2nd Lt USAF

AFIT/GA/MA/81D-1

A NEW METHOD OF SOLVING ILL-CONDITIONED SYSTEMS OF EQUATIONS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by
Brian S. Birmingham
2nd Lt USAF
Graduate Astronautical Engineering

December 1981

Preface

I should like to thank my thesis advisor, Dr. W. S. Ericksen, for his guidance and unbounded patience. When I strayed from the Ordained Path, he was a Shepherd, leading me back to the Fold.

Also, I extend thanks to my thesis committee, Dr. Quinn, Capt J.E. Marsh, and Dr. Paul Nikolai.

Brian S. Birmingham

TABLE OF CONTENTS

	Page
Preface.	ii
List of Tables	iv
Notations.	v
Abstract	vi
I. Introduction	1
II. Iterative Improvement.	5
III. Masking.	7
IV. Methods.	8
V. Results.	10
Given $\bar{x}$, obtain $\bar{b}$, and solve $A\bar{x} = \bar{b}$	10
For the system $A\bar{x}=\bar{b}$, given A and $\bar{b}$, solve for $\bar{x}$.	14
VI. Recommendation	23
VII. Conclusions.	24
Bibliography	25
Appendix A: Derivation of Iterative Scheme and Convergance Condition	26

List of Tables

Table		Page
I	Effects of Iterative Improvement.	5
II	Effects of masking using a given $\bar{x}$, Case 1. . .	11
III	Effects of masking using a given $\bar{x}$, Case 2. . .	12
IV	Effects of masking using a given $\bar{x}$, Case 3. . .	13
V	Effects of masking using a given $\bar{x}$, Case 4. . .	14
VI	Residual errors for a given $\bar{b}$	19
VII	Residual errors with masking and a given $\bar{b}$. . .	20
VIII	Residual errors with masking and multiplier with a given $\bar{b}$	22
IX	Residual errors with masking and multiplier with a given $\bar{b}$ (matrix from ABMAT was used). .	23

Notation

A -	the matrix of coefficients in the system of equations, $Ax = b$
x -	the solution vector of the system, $Ax = b$
b -	the vector of constants on the right-hand side of a system of equation's
A1 -	the masked version of A
A2 -	the matrix obtained when A1 is subtracted from A

Abstract

The problems associated with ill-conditioned matrices are well known and widespread. As of yet, there are no general solutions to the problem. The only remedies are usually ad hoc and are extremely case-dependent.

This paper presents a method which gives good results for a large variety of situations. The method involves masking, a technique which sets the lower-order bits (the number of bits varies) to zero, and then applying standard system-of-equations solvers. Most of the computer runs made used the Hilbert matrix, and either a specified $\bar{b}$ vector (as in the form, $A\bar{x} = \bar{b}$), or a specified $\bar{x}$ vector, with the $\bar{b}$ obtained by multiplying $\bar{x}$ by A.

A NEW METHOD OF SOLVING ILL-CONDITIONED SYSTEMS OF EQUATIONS

by
Brian S. Birmingham
2nd Lt USAF
Dr. W. S. Ericksen, Advisor

"I have often admired the secret magic of numbers"
Sir Thomas Browne

Ill-conditioned matrices are a constant source of irritation for programmers, and provide much frustration for engineers and scientists who, even though they have modelled their particular physical system accurately, have no way of obtaining reasonable solutions from a system-of-equation solver.

The problem is not completely understood, yet it lies in the fact that small perturbations to the matrix elements result in large errors in the solution. Any matrix exhibiting this property is termed <u>ill-conditioned.</u>

Part of the problem lies with the floating-point number system. A computer word is a finite-length entity, and difficulties arise in trying to represent an infinite digit number with a finite number of digits. Many constants are irrational, such as π or e, but even rational numbers present problems, such as 2/3 or 4/7. Even with a large wordlength machine, the inaccuracies present can cause problems for all but the most well-conditioned systems. One must also consider numbers that are finitely represented in decimal, yet require an infinite number of digits in other number bases. Johnson and Riess give a good description of this [3]. For example, $.1_{10}$ is represented perfectly in base 10 with a finite number of digits. Yet in base 16, $.1_{10} = .19999\ldots\ldots_{16}$, and in binary, $.1_{10} = .001\ 1001\ 1001\ 1001\ 1001\ldots_{2}$.

Yet another problem with the floating-point number system is that of round off. There are three main causes of computer round-off error [7], all of which play a prominent role in matrix computation:

1. <u>The addition of Large Numbers to Small Numbers</u>

   If a particular matrix requires the addition of, say, 3.8×10^{12} plus 1.56×10^{-13}, the result will probably be 3.8×10^{12}, i.e., 1.56×10^{-13} did *not* show up. Granted, the error involved is small, yet for large systems the effects may build up. This is call <u>accumulation error</u>. One solution is to simply increase the wordlength, either by machine design or by the use of double precision. While this may give good results, the scheme still treats the symptoms and not the cause.

2. <u>Two almost Equal Numbers being Subtracted</u>

   For example, .123456789 - .123456788 =

.000000001. Before the operation we had 9 digits of accuracy. After the subtraction only one digit of accuracy is left. Many times, a situation like this will give an error that is several orders of magnitude bigger than the actual numbers.

3. Division by Small Number

Obviously, a division by 0 will result in an overflow. However, a division by a small number, say 10^{-14}, may also result in an overflow. There is another problem in that a number with only one or two significant digits.

The error in the floating-point number system can be determined. If x is a real number, we can represent it as,

$$x = \pm\ 0.n_1n_2n_3n_4 \ldots . n_k \times 10^e$$

where k depends on s, and can be infinitely large.

However, x is represented as an approximation in a finite wordlength machine as x', where

$$x' = \pm 0.n_1n_2n_3 \ldots n_s \times 10^e$$

where s is the precision of the machine.

The error involved is,

$$|x - x'| \leq \tfrac{1}{2} \times 10^{e-s}$$

Blum goes into this more rigorously [1], as does Knuth [4].

With the problem inherent in the floating-point number system, one might be tempted to give up completely any attempts at system-of-equation solving. Unfortunately, things get worse before they get (better, of course, is a relative term).

Another aspect should be discussed, that of condition numbers. Generally, the condition number of a matrix is used as an indicator of ill-conditioning. However, in order to obtain the condition number, one must first obtain the norm. The two most common vector norms are $||\bar{x}||_1$, and $||\bar{x}||_\infty$, called the one-norm and the two-norm, respectively. They are defined as,

$$||\bar{x}||_1 = \sum_{i=1}^{K} |x_i|$$

$$||\bar{x}||_\infty = \max_i |x_i|$$

The vector norm used in this study was the one-norm. The two matrix norms corresponding to the above are

$$||A||_1 = \max_j \sum_{i=1}^{K} |x_i|$$

which is simply the maximum absolute column sum. Also,

$$||A||_\infty = \max_i \sum_{i=1}^{K} |Aij| \text{(maximum absolute row sum)}$$

where A^H is the Hermitian transpose of A. This paper used the matrix one-norm. The condition number of A, denoted as cond(A), is now defined as

$$\text{cond}(A) = ||A|| \quad ||A^{-1}||$$

Immediately one can see that the condition number is best used as an a posteriori estimate, since the condition number involves computing the inverse of A.

It must, however, be kept in mind that a condition number is not an absolute measure of conditioning. Many matrices can exhibit large condition numbers, and still be well-conditioned. The Hilbert matrix, on the other hand, has a reasonable condition number (for the lower orders), yet is highly ill-conditioned. But generally, large condition numbers suggest ill-conditioning. There are many other norms and condition number, explained in detail in [5], and [6].

II. Iterative Improvement

While the typical user may face the Scylla of word-length constraints, and the Charybdis of condition number, he has a powerful tool at his disposal, that of iterative improvement.

The effects of iterative improvement are shown in the table below. The A is defined on page 9. P is a parameter used in defining A.

Table I

Effects of Iterative Improvement

		$P = 0.5$		$P = 0.5 - 10^{-3}$	
		Exact Floating-Point Elements		Non-Exact Floating-Point Elements	
		Non-improved	Improved	Non-improved	Improved
k	cond(A)	$\frac{\|\|B1-B\|\|}{\|\|B\|\|}$	$\frac{\|\|B2-B\|\|}{\|\|B\|\|}$	$\frac{\|\|B1-B\|\|}{\|\|B\|\|}$	$\frac{\|\|B2-B\|\|}{\|\|B\|\|}$
2	2.5 + 03	0.0	0.0	1.9 - 13	3.6 - 14
4	4.9 + 08	2.6 - 11	0.0	7.8 - 11	7.9 - 11
6	3.2 + 13	2.3 - 08	0.0	1.4 - 07	8.8 - 08
8	1.2 + 18	3.1 - 04	0.0	2.0 - 04	9.0 - 05
10	3.2 + 22	3.8 - 01	0.0	1.3 - 01	2.9 - 01

where

$B = A^{-1}$

$N = 16$

$B1 = A^{-1}$ from LINV1F

$B2 = A^{-1}$ from LINV2F

LINV1F and LINV2F are routines in the IMSL software library.

The concept involves improvement upon the obtained solution. Generally, the improvement is focused on the residuals, where

$\bar{r} = A\bar{x} - \bar{b}$

Hopefully, $\bar{r} = \bar{0}$, but usually, because of conditions previously discussed, there will be error. Typically one would proceed:

Solve $A\bar{x} = \bar{b}$,

compute $\bar{r} = A\bar{x} - \bar{b}$,

Solve $A\bar{x}' = \bar{b} - \bar{r}$,

Solve $A\bar{y} = \bar{r}$,

and obtain,

$\bar{y} = \bar{x} - \bar{x}'$

and thus obtaining a correction for $\bar{x}$. This process is repeated until $||\bar{y}||<\varepsilon$, where ε is a user-defined tolerance.

Another form of iterative improvement involves splitting, whereby A is split into two matrices such that,

$A = M + N$

The iterative process then takes the form,

$M\bar{x}^{i+1} = N\bar{x}^{i} + \bar{b}$

and the initial $\bar{x}$ is obtained by solving

$$M\bar{x}^0 = \bar{b}$$

The only requirement is that M^{-1} exist.

III. Masking

The method pursued here involved masking (a variation of splitting), which essentially sets a number of lower order bits equal to zero. The number of bits set equal to zero varied.

For example, if one wanted to use a mask of 6 bits, i.e., set the last 6 bits equal to zero, the mask would be defined as

$$\text{MASK} = 77777........7700_8$$

and then each element of the matrix would be put through the AND function with MASK. This process would keep the exponent and the significant portion of the mantissa untouched, while setting the last 6 bits equal to zero. For coding purposes, simply set MASK = -77_8 and excute the following (CDC convention):

```
   DO 10 I = 1,K
   DO 20 J = 1,K
   A1(I,J) = A(I,J) .AND. MASK
20 CONTINUE
10 CONTINUE
   DO 30 I = 1,K
   DO 40 J = 1,K
```

```
A2(I,J) = A(I,J) - A1(I,J)
40 CONTINUE
30 CONTINUE
```

where K is the order of the matrix, A1 is the masked version of A and

$$A2 = A - A1$$

IV. Methods

The methods of dealing with the solution of $A\bar{x} = \bar{b}$ were used in this study. The first involved specifying a solution vector $\bar{x}$, then multiplying by A to obtain $\bar{b}$. Then A and $\bar{b}$ were sent to a system-of-equation solver (LEQT2F from the IMSL package, and SGEIM from the LINPACK package). For a control run, the complete (unmasked) A and $\bar{b}$ were used and the results noted. Then A was masked and $\bar{b}$ computed, using the same $\bar{x}$. Then the masked A, called A1, and the resulting $\bar{b}$, called $\bar{b}'$, were sent to the system-of-equation solver.

The second problem involved specifying a $\bar{b}$, and sending A and $\bar{b}$ to the solver. A form of iterative improvement was used after each return from the solver, namely,

$$A1\bar{x}^{i+1} = \bar{b} - A2\bar{x}^{i}$$

and the initial $\bar{x}$ was obtained by solving,

$$A1\bar{x} = \bar{b}.$$

The derivation and covergence conditions are contained in Appendix A.

For the first case, an error check consisted of $||\bar{x}_{given} - \bar{x}_{obtained}||$, where the norm was the sum of the absolute values of each element.

For the second case, the iteration was stopped when $||\bar{x}^{i+1} - \bar{x}^{i}|| = 0$. For final error estimates, a check on the residuals was used,

$$\text{error} = \frac{||Ax - b||}{||b||}$$

When b was specified, there was no iteration on the solution when a mask was not used, since A2 = [0].

For most of the results, the Hilbert matrix was used. The Hilbert is very easy to construct and has the added bonus of all elements being integers. The other matrix used came from the AFIT subroutine, ABMAT. It is constructed using the following algorithm:

$$A_{1j} = P \qquad J = 1,K$$

$$A_{i1} = \begin{cases} \binom{N}{i-1} = \dfrac{N!}{(i-1)!\,(N-i+1)!} & i = 2,K \text{ provided } i-1 \leq N \\ 0. \text{ if } i-1>N \end{cases}$$

$$A_{ij} = A_{i,j-1} + A_{i-1,j-1} \qquad i,j = 2,K$$

where P is any real, non-zero number and N is any non-negative integer. As in the Hilbert, this matris is easy to obtain, the inverse is easy to obtain, and ill-conditioning is easily obtained by specifying P to be some rational number plus a small perturbation, such as

$$P = .5 + 10^{-13}$$

V. Results

All of the results generated here were obtained on the CDC Cyber 74, which has a 60 bit word. The Operating System is the NOS/BE, and the language used was Fortran 4. The default, no rounding, was used in all computations.

A: Given $\bar{x}$, obtain $\bar{b}$, solve $A\bar{x} = \bar{b}$. This method used a given $\bar{x}$, which was multiplied by A to obtain $\bar{b}$, and then A and $\bar{b}$ were sent to the system-of-equation solver. Although this in not a "real world" example, this method is useful since it gives an absolute check on the solution obtained. Residual checking is not needed since we know what the solution must be. All runs in this section were made with LEQT2F, and the Hilbert matrix.

For a control, the straight (unmasked) A and corresponding $\bar{b}$ were sent to LEQT2F. For the masked case, A was masked first, then $\bar{x}$ multiplied by A to obtain $\bar{b}'$, and then A and $\bar{b}$ were sent to LEQT2F. Note the $\bar{b}'$ will differ from $\bar{b}$. The following table gives the lowest error for a given order, with

$$\bar{x}^T = [1.1.1.......1.]$$

Table II

Effects of masking using a given $\bar{x}$, Case 1

Order	Straight Error	Masked Error
5	.97053E-09	.14219E-13
10	.19547E-01	0.0
15		.21316E-13
20		.71454E-14
25		.71454E-14
30		0.0
35		0.0
40		.21316E-13
45		.28422E-13
50		.28422E-13

Although the unmasked system failed to converge after K = 12, the masked version did converge, up to K = 50, and the errors are remarkable. Perhaps a table of condition numbers would help to emphasize:

ORDER	CONDITION NUMBER
5	.94366E+06
10	.35357E+14
15	.15393E+22
20	.62836E+29
25	.27745E+37
30	.11777E+45
35	.51856E+52
40	.22451E+60
45	.98823E+67
50	.43303E+75

Even with the condition number present for K = 50, the masked solution converged.

Next, an $\bar{x}$ of

$$\bar{x} = \begin{bmatrix} 10.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ 1.0 \end{bmatrix}$$

was used, and the table now gives

Table III

Effects of masking using a given $\bar{x}$, Case 2

Order	Straight Error	Masked Error
5	.32923E-08	.71054E-13
10	.74683E-01	0.0
15		.28422E-13
20		0.0
25		.78160E-13
30		.49738E-13
35		.63949E-13
40		.78159E-13
45		.92370E-13
50		.42635E-13

As before, the unmasked solution failed to converge for orders greater than 11, yet the masked solution converged for all orders up to 50.

Now an $\bar{x}$ of

$$\bar{x} = \begin{bmatrix} 10.0 \\ 10.0 \\ 10.0 \\ \vdots \\ . \\ . \\ 10.0 \end{bmatrix}$$

was used and the results are:

Table IV

Effects of masking using a given $\bar{x}$, Case 3

Order	Straight Error	Masked Error
5	.46413E-08	.11369E-12
10	.64939E+00	.28422E-13
15		.39790E-12
20		.56843E-12
25		.56843E-12.
30		.79581E-12
35		.90949E-12
40		.96634E-12
45		.10800E-11
50		.11937E-11

Similar results were obtained.

Finally, an $\bar{x}$ of

$$\bar{x} = \begin{bmatrix} 1.0 \\ 2.0 \\ 3.0 \\ 4.0 \\ . \\ . \\ . \\ k.0 \end{bmatrix}$$

was used and the results are:

Table V

Effects of masking using a given $\bar{x}$, Case 4

Order	Straight Error	Masked Error
5	.28456E-09	0.0
10	.43743E-01	.11368E-12
15		.19895E-12
20		.31974E-12
25		.85265E-12
30		.11013E-11
35		.11795E-11
40		.20037E-11
45		.22879E-11
50		.31690E-11

Once again, these results are remarkable and have been verified by W. Ericksen.

B. For the system $A\bar{x} = \bar{b}$, given A and $\bar{b}$, solve for $\bar{x}$. This method is a good test problem because the typical user will have at his disposal the A matrix and the $\bar{b}$ vector, with no knowledge of $\bar{x}$, or at best, an idea of the range of values in $\bar{x}$.

For the first part of this section, the Hilbert was used for reasons given previously. The choice of $\bar{b}$ was made so as to give the system-of-equation solver the worst possible situation.

For this reason, we chose,

$$\bar{b} = \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ \vdots \\ \cdot \\ \cdot \\ 0.0 \end{bmatrix}$$

where the length of $\bar{b}$ depended on the order of the system. By choosing this particular vector, the solution, $\bar{x}$, will be the first column of the inverse of A. This is one of the hardest cases to solve since a system-of-equation solver typically does not solve for the inverse, and this $\bar{b}$ will force it to do so. Also, the norm of $\bar{x}$ will be quite large, as shown later. Since the first column of the inverse a A will be needed, the choice of the Hilbert matrix is convenient because of the ease of computing the inverse Hilbert. Physically, the $\bar{b}$ chosen could represent a unit impulse applied to only one input.

The program flow can be described heuristically:

1). Obtain A and A^{-1} (A^{-1} used for computation of cond(A))

2). Set $\bar{b} = 0$ and $\bar{b}(1) = 1$.

3). Solve $A\bar{x} = \bar{b}$ and save $\bar{x}$

4). Apply a mask to A and solve $A1\bar{x} = \bar{b}$, using the interative procedure, $A1\bar{x}^{i+1} = \bar{b} - A2\bar{x}^{i}$, stopping when convergence is satisfied.

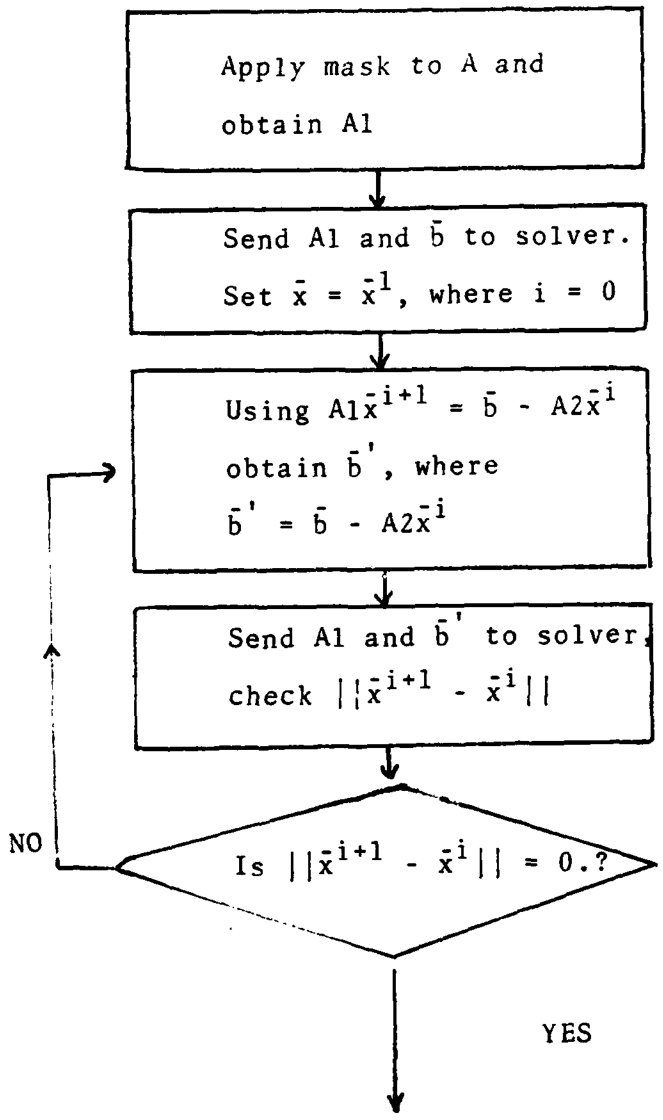
5). Compare results obtained from 3) and 4)

6)a. Apply different mask and repeat 4) and 5)

b. if all masks have been applied, increment order of matrix and go to 1).

Step 1) is easily implemented by a call to a subroutine and giving as input the order desired. Step 2) is straightforward. Step 3) is merely a call to a system-or-equation solver, and inputting A and $\bar{b}$, along with appropriate work matrices and parameters. The primary solver used was LEQT2F. LEQT2F applies iterative improvement internally. SGEIM was used intermittently. SGEIM also uses internal iterative improvement.

Step 4) can best be described with a "pseudo" flowchart:



Apply mask to A and obtain A1
Send A1 and $\bar{b}$ to solver. Set $\bar{x} = \bar{x}^1$, where i = 0
Using $A1\bar{x}^{i+1} = \bar{b} - A2\bar{x}^i$ obtain $\bar{b}'$, where $\bar{b}' = \bar{b} - A2\bar{x}^i$
Send A1 and $\bar{b}'$ to solver, check $||\bar{x}^{i+1} - \bar{x}^i||$
Is $||\bar{x}^{i+1} - \bar{x}^i|| = 0.?$
NO
YES

Step 5)

The norm used for convergence checks was simply,

$$||\bar{x}^{i+1} - \bar{x}^i|| = \sum_{j=1}^{K} |x_j^{i+1} - x_j^i|$$

The comparison in step 5) was performed using residuals, i.e.,

$$\text{error} = \frac{||A\bar{x} - \bar{b}||}{||b||}$$

The error with masking used the $\bar{x}$ obtained from step 3), and the error with masking used the $\bar{x}$ obtained from step 4).

Step 6)a simply incremented the mask applied. The masks used were, in order,

-0_8, -1_8, -3_8, -7_8, -17_8, -37_8, -77_8,
-777_8, -7777_8, -77777_8, -777777_8, -7777777_8.

These correspond to the number of bits set equal to zero, respectively,

0 bits, 1 bits, 2 bits, 3 bits, 4 bits, 5 bits, 6 bits, 9 bits, 12 bits, 15 bits, 18 bits, 21 bits.

After each incrementation of the mask, the flow proceeded to step 4), since solving the original system, $A\bar{x} = \bar{b}$, would give identical results.

After the last mask had been applied, the order of the matrix was incremented in step 6)b, and the program began a new at step 1). The program was stopped at an order of 12, since LEQT2F and SGEIM failed to converge while solving $A\bar{x} = \bar{b}$. Although error estimates could have been obtained, the results might possibly be faulty and misleading.

The error in the iterative loop, or the convergence criteria in step 4) was forced to equal zero. In other words, the elements in the final solution vector were equal, on a bit-by-bit basis, to the elements in the preceding solution vector. This constraint presented no problems in convergence. However, several times the solution failed to converge after 30 iterations. Convergence could only be obtained by letting the tolerance assume unacceptable values.

As a comparison, the unmasked Hilbert and a $\bar{b}$ of all zeroes, with b(1) = 1 was used:

Table VI

Residual errors for a given $\bar{b}$

ORDER	RESIDUAL ERROR (LEQT1F)	RESIDUAL ERROR (LEQT2F)	RESIDUAL ERROR (SGEIM)
4	.75351E-12	.69972E-13	.69972E-13
5	.37334E-11	.17584E-11	.17584E-11
6	.17141E-10	.39022E-10	.39022E-11
7	.82591E-10	.67809E-10	.67809E-10
8	.79065E-09	.70000E-09	.70000E-09
9	.55734E-08	.21301E-08	.21301E-08
10	.51631E-07	.36709E-07	.36709E-07
11	.16794E-06	.90699E-07	.90699E-07

LEQT1F had bigger errors due to lack of iterative improvement.

Now masking was implemented, using the same $\bar{b}$:

Table VII

Residual errors with masking and a given $\bar{b}$

ORDER	RESIDUAL ERROR (LEQT1F)	RESIDUAL ERROR (LEQT2F)	MASK
4	.69972E-13	.69972E-13	-1_8
5	.87555E-12	.87555E-12	-7777777_8
6	.33090E-11	.33090E-11	-7777777_8
7	.40057E-10	.40057E-10	-7777_8
8	.48427E-09	.48427E-09	-7777_8
9	.14453E-08	.14453E-08	-7777_8
10	.28956E-07	.28956E-07	-77_8
11	.90699E-07	.90699E-07	-1_8

Notice that masking improved the error for almost every order chosen.

The solution vector for this section is simply the first column of the inverse Hilbert. The solution vector for order 10 is:

$$\bar{x} = \begin{Bmatrix} 100 \\ -4950 \\ 79200 \\ -600600 \\ 2522520 \\ -6306300 \\ 9609600 \\ -8751600 \\ 4375800 \\ -923780 \end{Bmatrix}$$

The norm of this vector is very large. However, the large norm of $\bar{x}$ motivated the use of a multiplier, namely,

$$\bar{b} = \left\{ \begin{array}{c} V \\ 0.0 \\ 0.0 \\ 0.0 \\ \cdot \\ \cdot \\ \cdot \\ 0.0 \end{array} \right\}$$

By choosing V to be, say, 10^{-5}, the solution vector will be smaller, thereby bringing the solution vector closer to a smaller norm. Now the method involved choosing an order and a multiplier and cycling through all of the masks. Then, the multiplier was incremented and the process repeated. The range of the multipliers were 10^{-8} through 1. The following table gives the best error obtained for a given order. For comparison purposes, some of the previous results are given:

Table VIII

Residual errors with masking and multiplier with a given S

ORDER	RESIDUAL ERROR (LEQT1F) (no masking)	RESIDUAL ERROR (LEQT2F) (masking only)	RESIDUAL ERROR (LEQT2F) (mask & mult)	MASK	MULT
4	.69972E-13	.69972E-13	.96857E-14	-7777777_8	10^{-4}
5	.17584E-11	.87555E-12	.18736E-12	-777_8	10^{-2}
6	.39022E-10	.33090E-11	.35767E-12	-7777777_8	10^{-7}
7	.67809E-10	.40057E-10	.49903E-11	-7777777_8	10^{-5}
8	.70000E-09	.48427E-09	.24875E-10	-7777_8	10^{-5}
9	.21301E-08	.14453E-08	.89556E-10	-7777_8	10^{-3}
10	.36709E-07	.28956E-07	.85877E-09	-17_8	0.1
11	.90699E-07	.90699E-07	.43302E-08	-3_8	0.1

While masking gives improved results, the results using a multiplier are even better.

As explained in Appendix A, convergence is satisfied if $||A1^{-1}A2||<1$. A check was made for all masks and convergence was satisfied.

A different A was now used, obtained from ABMAT, as previously discussed. An N of N = 16 and a $P = .5 + 10^{-13}$ was chosen. The following table gives results for no masking, masking, and masking with a multiplier:

Table IX

Residual errors with masking and multiplier with a given 5

(matrix from ABMAT was used)

ORDER	RESIDUAL ERROR (LEQT1F) (no masking)	RESIDUAL ERROR (LEQT2F) (masking only)	RESIDUAL ERROR (LEQT2F) (mask & mult)	MASK	MULT
4	.37255E-08	.37255E-08	.37255E-08	-7777777_8	1.0
5	.61402E-04	.52083E-04	.11945E-05	-3_8	10^{-6}
6	.10688E-01	.10688E-01	.83189E-04	-17_8	10^{-4}
7	.96037E+00	.59864E+00	.90140E-01	-17_8	10^{-2}
8	.46178E+02	.46178E+02	.15700E+02	-17_8	10^{-8}

The results stopped at order 8 because LEQT2F failed to converge for higher orders. These results have been verified by W. Ericksen.

CPU times have not been given due to the amount of extraneous computation.

VI. Recommendations

All of the results were obtained with a 60-bit machine. Other machines should be used. Possible, the algorithm devised by Brent [2], could be implemented, since it can simulate any wordlength desired. Also, the results obtained in Va should be investigated more thoroughly, perhaps by trying different $\bar{x}$ vectors and using other matrices.

VII. Conclusion

The results obtained show masking to be quite useful in ill-conditioned systems when an A and $\bar{b}$ are specified, and $\bar{x}$ not known. However, for some cases, improvement was negligible. Masking is easily implemented, and can provide good results for a minimum increase in computer time.

The results of Va are quite spectacular, and while not completely realistic, they are important enough to warrant further investigation. They show that for small changes in A and $\bar{b}$, convergence is obtained where it was previously impossible.

"As yet a child, nor yet a Fool to Fame,
I lisp'd in numbers, for the numbers came"
Alexander Pope

Bibliography

1. Blum, R. Numerical Analysis and Computations Theory and Practice.
2. Brent, R. ACM Transactions on Mathematical Software, March 1978.
3. Johnson, Lee W. and R. Dean Riess. Numerical Analysis.
4. Knuth, Donald. The Art of Computer Programming, Vol. 2.
5. Noble, Ben. Applied Linear Algebra.
6. Rice, John. Matrix Computations and Mathematical Software.
7. Steinberg, David. Computational Matrix Algebra.

Appendix A

Let the system be defined as

$$A\bar{x} = \bar{b}$$

and let

$$A = A1 + A2$$

where

$$A1 = A(\text{masked})$$

Now the system becomes

$$(A1 + A2)\bar{x} = b$$

$$A1\bar{x} + A2\bar{x} = \bar{b}$$

$$A1\bar{x} = \bar{b} - A2\bar{x}$$

Since A2 has only small elements, our iteration can proceed as

$$A1\bar{x}^{i+1} = \bar{b} - A2\bar{x}^{i}$$

where $\bar{x}_{initial}$ is obtained by solving $A1\bar{x} = \bar{b}$

For convergence,

$$A1(\bar{x}-\bar{x}_1) = -A2(\bar{x}-\bar{x}_0)$$

$$(\bar{x}-\bar{x}_1) + -A1^{-1}A2(\bar{x}-\bar{x}_0)$$

$$A1(\bar{x}-\bar{x}_2) = -A2(\bar{x}-\bar{x}_1) = A2A1^{-1}A2(\bar{x}-\bar{x}_0)$$

$$(\bar{x}-\bar{x}_2) = -A1^{-1}A2(\bar{x}-\bar{x}_1)$$

$$A1(\bar{x}-\bar{x}_3) = -A2(\bar{x}-\bar{x}_2). = (-A2)(-A1^{-1}(\bar{x}-\bar{x}_1))$$

$$= (-A2)(-A1^{-1}A2)(-A1^{-1})(\bar{x}-\bar{x}_0)$$

$$= (-A2)(-A1^{-1}A2)^2(\bar{x}-\bar{x}_0)$$

$$(\bar{x}-\bar{x}_3) = (-A1^{-1})^3(\bar{x}-\bar{x}_0)$$

so,

$$\bar{x}-\bar{x}_n = (-A1^{-1}A2)^n(\bar{x}-\bar{x}_0)$$

$$||\bar{x}-\bar{x}_n|| < ||-A1^{-1}A2||^n \; ||\bar{x}-\bar{x}_0||$$

$$||-A1^{-1}A2||^n < 1.$$

$$-||A1^{-1}A2||^n < 1.$$

since $||A1^{-1}A2||$ is always greater than zero

$||A1^{-1}A2|| < 1.$ for convergence.

These derivations are due to W. Ericksen.

VITA

Brian S. Birmingham was born in Memphis, Tennessee on June 15, 1958. He graduated from Harding Academy in May of 1976. He attended Mississippi State University and graduated in May of 1980 with a Bachelor of Science Degree in Aerospace Engineering. He began his studies at the Air Force Institute of Technology in June of 1980, majoring in Astronautical Engineering.

Permanent Address: 10730 E. Goodman
Olive Branch, MS 38654

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GA/MA/81D-1	2. GOVT ACCESSION NO. AD-A115 569	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A NEW METHOD OF SOLVING ILL-CONDITIONED SYSTEMS OF EQUATIONS		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Brian S. Birmingham		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology(AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE December 1981
		13. NUMBER OF PAGES 37
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

15 APR 1982

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17.

[illegible]

~~F. C. LYNCH, Major,~~ USAF
~~Director of Inform~~ation

Dean for Research and
Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The problems associated will ill-conditioned matrices are well known and widespread. As of yet, there are no general solutions to the problem. The only remedies are usually ad hoc and are extremely case-dependent.
This paper presents a method which gives good results for a large variety of situations. The method involves masking, a technique which sets the lower-order bits (the number of bits varies) to zero, and then applying standard (continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

system-of-equations solvers. Most of the computer runs made used the Hilbert matrix, and either a specified $\bar{b}$ vector (as in the form, $A\bar{x} = \bar{b}$), or a specified $\bar{x}$ vector, with the $\bar{b}$ obtained by multiplying $\bar{x}$ by A.

LMED

8